VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2014 Zbyněk Šafarčík, DiS.

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Zbyněk Šafarčík, DiS.**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: EVici webdesign s.r.o.
2. Struktura závěrečné zprávy:
a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta
b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti
c) Zvolený postup řešení zadaných úkolů
d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe
e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe
f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vedl odbornou praxi studenta.

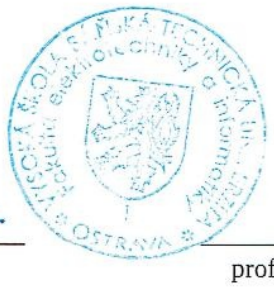
Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Miloš Kudělka, Ph.D.**

Konzultant bakalářské práce: Ing. Jan Rataj

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2014



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7.5.2014 ..... Zbyněk Šafařík

Rád bych na tomto místě poděkoval firmě EVici webdesign s.r.o. za možnost absolvovat zde praxi, a také všem kolegům za pomoc při řešení zadaných úkolů.

Abstrakt

Tato bakalářská práce se zabývá činností, kterou jsem vykonával ve firmě EVici webdesign s.r.o. Obsahuje představení firmy, popis technologií které používá a stručný popis úkolů, které jsem řešil. Dále obsahuje zhodnocení práce, chybějící a nabyté zkušenosti během praxe.

Klíčová slova: PHP, Nette, JavaScript, SQL, import, export

Abstract

This diploma thesis deals with the activity, what I performed in EVici webdesign s.r.o. It includes presentation of the company, description of the technologies used by company and a brief description of the tasks that I solved. It also includes assessment of the work, missing and learned skills during practice.

Keywords: PHP, Nette, JavaScript, SQL, import, export

Seznam použitých zkratek a symbolů

PHP	–	PHP: Hypertext Preprocessor
MVC	–	Model View Controller
MVP	–	Model View Presenter
XSS	–	Cross-site Scripting
CSRF	–	Cross-site Request Forgery
CMS	–	Content Management System
CRM	–	Customer Relationship Management
URL	–	Uniform Resource Locator
HTML	–	Hyper Text Markup Language
CSS	–	Cascading Style Sheets
AJAX	–	Asynchronous JavaScript and XML
XML	–	Extensible Markup Language
CSV	–	Comma-separated values
SQL	–	Structured Query Language
SOAP	–	Simple Object Access Protocol
WSDL	–	Web Services Description Language

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Individuální odbornou praxi jsem vykonával ve firmě EVici webdesign s.r.o. Zde jsem zastával funkci programátora PHP a frameworku Nette. Náplní mé práce bylo opravovat a ladit systém UPgates, programovat do něj nové moduly a funkce. Kromě programování v PHP jsem pracoval také s HTML, CSS, JavaScriptem a SQL.

V této bakalářské práci představím firmu EVici webdesign s.r.o., dále se zmíním o skriptovacím jazyku PHP a frameworku Nette. V další části se budu věnovat samotné práci, kterou jsem vykonával, jednotlivým úkolům, postupům a technologiím které jsem použil. Na závěr zhodnotím práci ve firmě, a jaký přínos pro mě práce měla.

2 Seznámení s firmou

Firma EVici webdesign s.r.o.[1] je na trhu od roku 2003, její prioritou je poskytování služeb v oblasti internetového marketingu a obchodu. Nabízí programování a poradenství pro online média, vyznačující se individuálním přístupem ke klientovi. Současný tým je tvořen pěti stálými pracovníky, firma sídlí v budově podnikatelského inkubátoru.

2.1 Produkty a služby

Firma se specializuje na tvorbu webových stránek, e-shopů, CMS a CRM. K tomuto účelu používají vlastní informační systém UPgates, který zahrnuje všechny tyto služby. UPgates je modulárním systémem, který poskytuje každému zákazníkovy přesně to, co potřebuje. Existují 3 základní stupně (web, web + CRM, web + eshop + CRM), které si zákazník může zakoupit. Liší se hlavně funkčností tzn. dostupnými moduly, a také cenou.

Z pohledu uživatele (zákazníka), zahrnuje UPgates dvě hlavní části a to je administrační rozhraní a tzv. „frontendová“ neboli veřejná část. Administrační rozhraní slouží pro editaci, správu obsahu a nastavení, „frontendová“ část je veřejně přístupná uživatelům.

Z pohledu aplikačního se jedná o dva oddělené systémy, které sdílejí některé knihovny. Aplikační logika je společná pro všechny zákazníky, projekty jednotlivých zákazníků se liší pouze v databázi a v designu veřejné části. To znamená, že když se provede změna v aplikaci například v části administrace, projeví se na všech projektech zákazníků. Dále pak existuje systémová databáze, která slouží pro ukládání informací o projektech, nastavení nebo importu produktů a dalších. Systém pak obsahuje ještě některé další oddělené části, jako je například import produktů, skripty pro Cron a další.

2.2 Technická řešení

Systém UPgates je vlastním produktem firmy EVici webdesign s.r.o. Je založen na skriptovacím jazyku PHP, frameworku Nette a dalších frameworcích (jQuery, Guzzle). K ukládání dat se používá databázový systém MySQL. UPgates je dynamický systém, který využívá většiny dostupných internetových technologií jako, jsou JavaScript, AJAX, FLASH, HTML5, CSS3.

Firma používá pro projekty svých zákazníků serverhosting, v prostorách firmy se nachází server, na kterém běží vývojová (testovací) verze systému UPgates.

[1] Více informací na http://www.evici.cz

3 PHP a Nette framework

3.1 PHP

PHP[2] je objektově orientovaný dynamicky typovaný skriptovací jazyk inspirovaný především jazyky Java a C++. Na jazyku PHP je založeno mnoho frameworků jako například Nette nebo Zend.

Mezi hlavní výhody patří rozsáhlý soubor funkcí, je multiplatformní, má podporu mnoha databázových systémů a také propracovanou a rozsáhlou dokumentaci.

Nevýhodou je, že PHP neudržuje kontext aplikace, musí jej vytvořit pokaždé znovu.

3.2 Nette framework

Nette[3] je open source framework, založený na jazyku PHP. Zaměřuje se především na bezpečnost a odstraňuje mnoho bezpečnostních rizik, jako jsou XSS, CSRF a další. Volitelnou součástí Nette je databázová vrstva Dibi a mnoho dalších komponent a rozšíření.

3.3 Ladění chyb

Nette framework odstraňuje některé nevýhody jazyka PHP, jako je například ladicí nástroj. Nejde jen o samotné lepší zobrazení a identifikace výjimky nebo chyby, ale také o dva režimy, produkční a vývojový. V produkčním režimu ladicí nástroj nezobrazuje výpis chyby, ale pouze zobrazí hlášení o chybě na straně serveru.

3.4 Architektura Nette

Nette používá architekturu MVP která je podobná MVC. Presenter zpracovává požadavky uživatele a na jejich základě vybírá view a předává mu data z modelu. Nette chápe URL odkaz jako volání funkce. Pod view si v Nette mohu představit šablonu uživatelského rozhraní.

[2] Více informací na http://www.php.net

[3] Více informací na http://www.nette.org

4 Práce

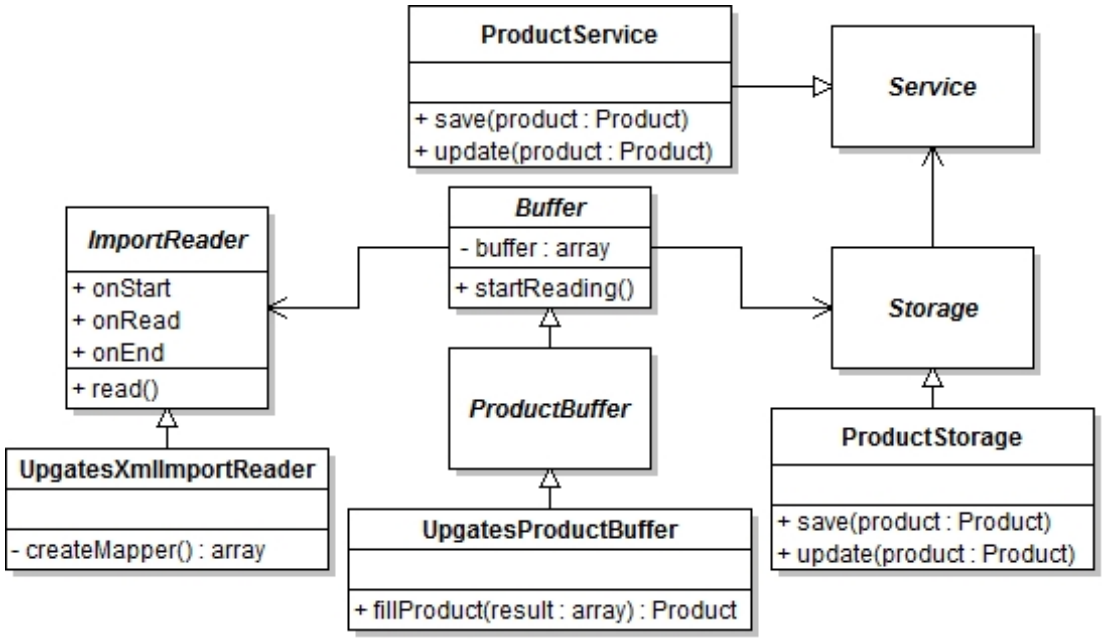
Moje pracovní zařazení bylo programátor PHP a frameworku Nette. V praxi to ale nebylo pouze psaní skriptů v PHP, ale také práce v HTML, CSS, JavaScriptu (jQuery) a SQL. Pro práci jsem používal vlastní notebook, dostal jsem přístup na server s testovací verzí systému UPgates. Mezi softwarové vybavení patřilo vývojové prostředí NetBeans, internetové prohlížeče Chrome, Firefox, Internet Explorer. Pracoval jsem hlavně ve Firefoxu, kde mám nainstalované rozšíření pro ladění chyb. Ostatní prohlížeče byly pouze pro testování.

Prvním úkolem bylo seznámit se se systémem. Nové moduly musely být vytvořeny při zachování stejné konvence jako ostatní moduly. Vzhledem k tomu že neexistuje žádná dokumentace k systému UPgates, musel jsem se řídit stávajícími moduly a v něčem mi pomohl firemní programátor.

Rozšíření funkcionalit a nové moduly se programovaly na základě potřeb a poptávky našich zákazníků a také z důvodu udržení kroku s konkurencí.

4.1 Import dat

Import je část systému, která se stará o naplnění systému daty. Práce zahrnovala doplňování o čtení a zpracovávání dalších dat o produktech, později rozšiřování architektury a tvorbu individuálních importů.

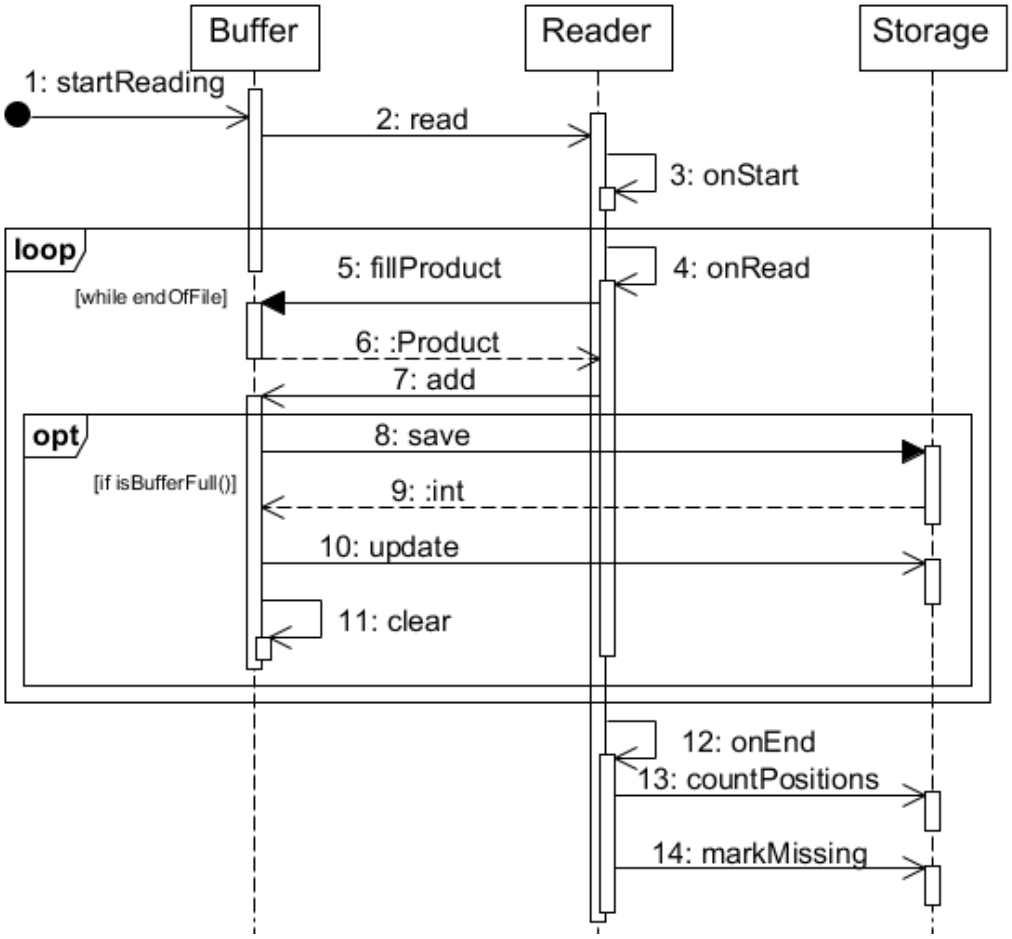


Obrázek 1: Třídní diagram - importu

Původní stav:

- možnost importovat pouze XML ve formátu pro systém UPgates
- pouze ruční vytvoření importu

Import využívá několik frameworků (Nette, Dibi, Guzzle, Symfony, Monolog) a má z pohledu architektury tři základní části, je to Reader, Buffer a Storage (obrázek 1 a 2). Mimo tyto základní objekty jsou zde ještě další objekty pro statistiky, nastavení, logování, připojení k databázi nebo stahování souborů.



Obrázek 2: Sekvenční diagram importu

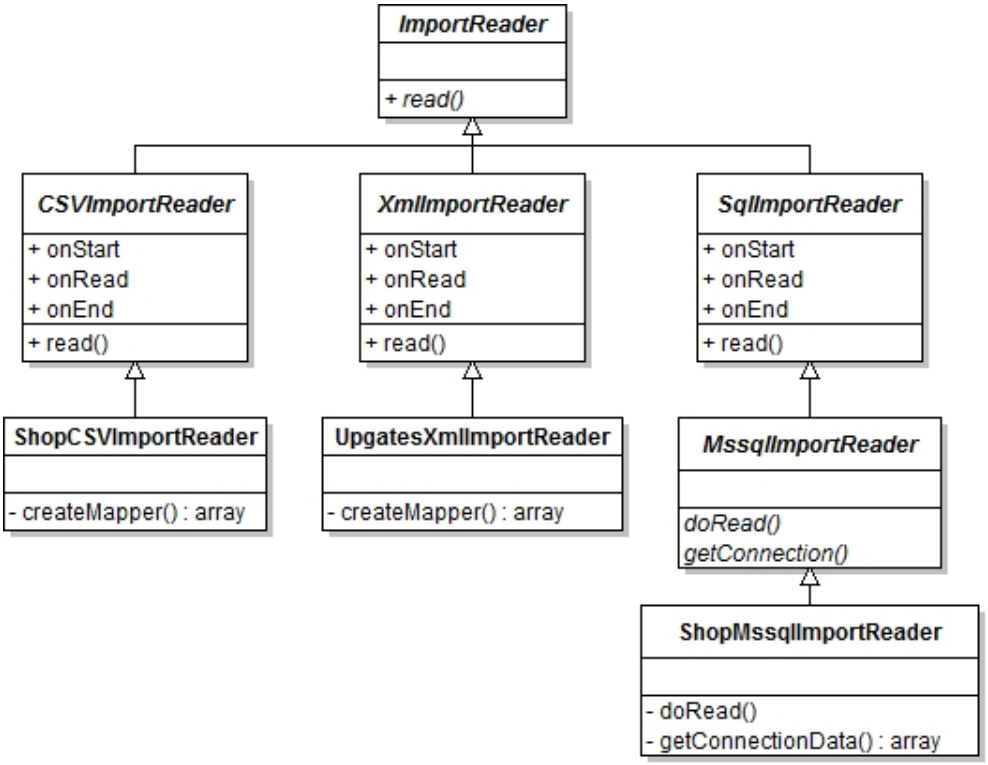
Při importu se z databáze načtou názvy konkrétních tříd pro konkrétní typ importu, a vytvoří se jejich instance.

Zpětná volání používaná při čtení:

- onStart - otevře soubor, je definován v konstruktoru konkrétní třídy pro Reader.

- onRead - volá se po přečtení každé položky, je definován v konstruktoru konkrétní třídy pro Buffer. Vytvoří objekt Product a předá ho na Buffer.
- onEnd - volá po ukončení čtení a provádí se operace, které není možné provést během importu, např. pozice produktů v kategorii. Volá metody na instanci konkrétní třídy Storage.

Takovýto návrh umožňuje definovat libovolný počet různých XML importů s různou strukturou. Stačí pouze implementovat konkrétní třídu pro Reader a Buffer.



Obrázek 3: Třídní diagram - upravený Reader

Mým úkolem bylo se seznámit se stávající architekturou a funkčností importu a postupně ji rozšiřovat. Import dosud uměl číst pouze XML ve formátu pro systém UPgates. Několik zákazníků potřebovalo importovat data z vlastních individuálních XML. To jsem realizoval implementací dalších tříd pro Reader a Buffer. Ve třídě pro Reader jsem implementoval vždy metodu createMapper, a ve třídě pro Buffer vždy zpětné volání onRead. Pak jsem přidal záznamy do databáze odpovídající novému typu importu (názvy tříd a nastavení).

Dalším individuálním importem produktů bylo čtení dat z CSV souboru. Se systémem v tomto stavu nebylo možné přidat jednoduše čtení z CSV, a proto jsem provedl změny ve třídách pro Reader (obrázek 3). Metoda read v každém z Readeru implementuje čtení a volá zpětná volání onStart, onRead, onEnd.

Nyní bylo možné přidat jakýkoliv další způsob čtení dat, například z databáze SQL (obrázek 3).

Zpětná volání používaná při čtení z SQL:

- onStart - připojí se do databáze za pomocí přístupových dat (IP adresy, hesla, atd.) které mu vrátí metoda getConnectionData.
- doRead - metoda řeší konkrétní čtení dat z databáze, volá zpětné volání onRead

Opět jsem se snažil navrhnout do budoucna maximálně rozšiřitelnou architekturu, v případě nutnosti vytvořím další třídu a implementuji odpovídající metody pro čtení a vše bude fungovat.

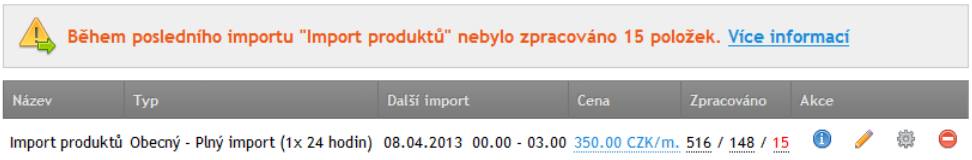
Toho jsem ihned využil, na základě požadavku zákazníka jsem rozšířil import o možnost číst data přes službu SOAP, a to implementací konkrétních tříd pro Reader a Buffer. K tomu jsem použil třídu Client z frameworku Zend, která umí se SOAP pracovat. Zend framework nebyl součástí importu, proto jsem ho musel přidat.

```
require_once LIBS_DIR . "/SOAP/vendor/autoload.php";
$client = new \Zend\Soap\Client(LIBS_DIR . "/SOAP/WSDL/WSDL.wsdl");
$params = array("login"    => "***",
                "password" => "***"
          );
$result = $client->KompletniKatalog($params);
```

Výpis 1: Použití třídy Client

Potřeboval jsem ale jen třídu Client. Protože nelze z frameworku zkopírovat jen jednu třídu z důvodu možných závislostí na dalších částech frameworku, použil jsem Composer. Je to balíčkovací systém pro PHP a jeho pomocí jsem vytvořil balíček obsahující třídu Client a všechny závislosti. Po přilinkování balíčku stačilo zavolat metodu SOAP služby na instanci třídy Client. V konstruktoru třídy Client předávám cestu k WSDL souboru, což je dokument založený na XML a popisující konkrétní službu SOAP (výpis 1). Metoda vrací pole objektů s daty o produktech. Pak už je vše podobné jako v ostatních typech importu.

Další součástí importu, kterou jsem měl za úkol vytvořit je skript pro Cron. Ten vybere z databáze importy a spustí je jako podproces. Při větším množství importů je neefektivní spouštět importy za sebou ve frontě, proto jsem přidal možnost paralelního spouštění.



Obrázek 4: Import v administraci

V tomto stavu byl import pro administrátora systému (našeho zákazníka) neviditelný, nevěděl kolik má importů, kolik se vytvořilo nových položek a kolik se jich aktualizovalo,

a navíc se každý nový import se musel ručně vytvořit v databázi. Mým dalším úkolem bylo vytvořit administrační rozhraní, ve kterém by si administrátor vytvořil import a viděl by čas importu, počet zpracovaných položek a další informace. Administrační rozhraní (presenter a šablona) již existovalo, jen na něm byla pouze dokumentace k obecnému importu. V administraci je tabulka importů a standardní operace přidání, editace a smazání importu (obrázek 4).

Cílem bylo:

- rozšířit import o možnost čtení dat z dalších zdrojů (CSV, SOAP, SQL)
- dopracovat import část Storage, zpracovávání dat a ukládání do databáze
- zajistit automatické spouštění importů - vytvořit skript pro Cron
- přidat do administrace správu importů, s možností zobrazení statistik, editaci atd.

Úkoly související s importem jsem strávil přibližně 30 dní práce.

4.2 Export dat

Mým dalším úkolem bylo vytvořit v systému komponentu, která se bude starat o export dat.

Původní stav:

- možnost exportu pouze omezeného množství vybraných objednávek
- žádná další možnost exportu dat

Výstupem mělo být XML a CSV soubor. Vytvoření a nastavení exportu probíhá v administraci, jsou k dispozici dva druhy exportu:

- okamžitý export – pouze některé položky, výstup ihned k dispozici (třída ExportEntity)
- plánovaný export – mnoho dat, náročný na provedení, jednou za den (třída ExportEntityCron), přístupný přes URL

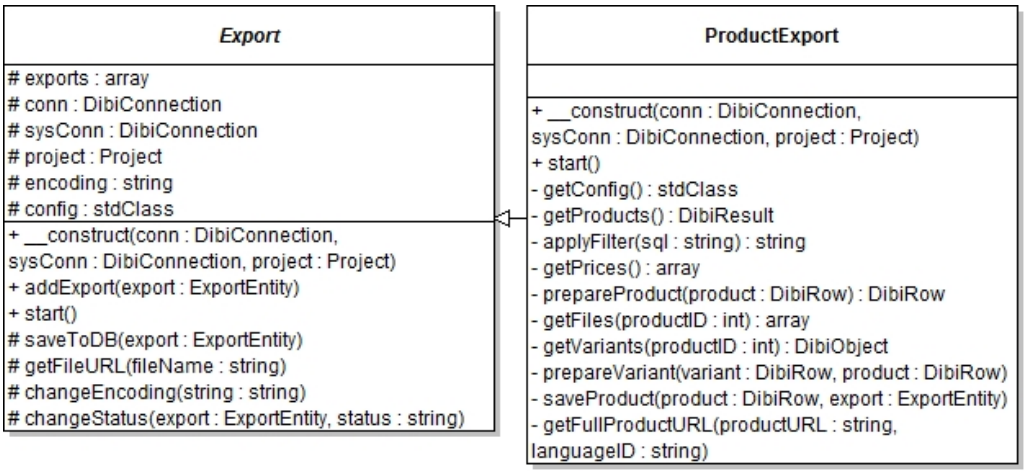
Pro vytváření exportů používám třídu AppExport. Kdekoliv v systému si vytvořím její instanci a mohu založit nový export (výpis 2).

```
$appExport = new AppExport(dibi::getConnection());
$type = $appExport->getExportType("contacts", "xml", true);
$appExport->makeNewExport($project_id, $type, array("only" => $customers));
```

Výpis 2: Použití třídy AppExport

Pro export vytvořím instanci odpovídající třídy, například ProductExport (obrázek 5), předám instance ExportEntity.

- ProductExport, CategoryExport, OrderExport, ContactsExport – vybírají a zpracovávají data
- ExportEntity – zapouzdřuje jeden export, jeho nastavení (okamžitý export)
- ExportEntityCron – dědí z ExportEntity, navíc ví kam má soubor uložit, a o který export v databázi se jedná



Obrázek 5: Třída ExportEntity

Metody třídy ProductExport:

- getConfig - projde všechny objekty ExportEntity, vytvoří sjednocení všech nastavení, cílem je vybrat z databáze položky jen jednou (pro všechny exporty jednoho typu položek)
- getProducts – vybere z databáze všechny produkty dle nastavení
- prepareProduct – načte z databáze ostatní informace o produktu, volá se vždy nad jedním produktem
- saveProduct – uloží produkt do XML

Podobně jako v importu je součástí exportu i skript pro Cron. Ten opět vybere z databáze exporty a provede je.

Cílem bylo:

- vytvořit export pro produkty, zákazníky, kategorie a začlenit do něj export objednávek
- umožnit okamžitý export, kdy výsledný soubor je ihned k dispozici
- naplánovaný export většího množství dat pomocí skriptu pro Cron

Úkoly související s exportem jsem strávil přibližně 20 dní práce.

5 Závěr

V průběhu vykonávání praxe ve firmě EVici webdesign s.r.o. jsem se naučil mnoho nových postupů a dovedností. Prohloubil jsem si znalost Nette frameworku, jazyka PHP, SQL a dalších technologií s nimiž jsem pracoval. Naučil jsem se hlavně si práci předem rozmyslet a programovat jednoduše, efektivně a univerzálně s ohledem na další rozšiřování funkcionality a možnosti systému.

Při mé práci mi hlavně zpočátku scházely hlubší znalosti Nette, objektového přístupu k programování a také JavaScriptu. Znalosti jsem získával postupně řešením aktuálních problémů a úkolů, snažil jsem se problémy prodiskutovat s kolegy, zda je takové řešení správné a také vycházet z dokumentace k dané technologii.